Réécriture de graphes

Dominique Duval

Grenoble, le 12 février 2009
Séminaire LJK-MAD BIPOP-CASYS

Outline

Introduction

Réécriture de termes

Réécriture de graphes

Conclusion

- Rachid Echahed et Frédéric Prost :
  réécriture de graphes “algorithmique”
- Transformation de graphes :
  passage au cadre catégorique
- D’où cette collaboration
  fait marquant du LJK 2007-08

Outline

Introduction

Réécriture de termes

Réécriture de graphes

Conclusion

Système de réduction

Un système de réduction c'est :

- un ensemble A d'objets
  Ex. L'ensemble des termes
  sur la grammaire $t ::= 0|1|2|\ldots|t+t|t-t$
  $(2+4)\times(5+1) \neq 6\times(5+1) \neq 6\times 6 \neq 36$
- et une relation binaire $\rightsquigarrow$ sur A
  Ex. $G \rightsquigarrow H$ *ssi*
  on passe de G à H en effectuant une opération
  $(2+4)\times(5+1) \rightsquigarrow 6\times(5+1) \rightsquigarrow 6\times 6 \rightsquigarrow 36$

Relations associée à un système de réduction

Soit A avec $\rightsquigarrow$ un système de réduction.

$(2+4)\times(5+1) \rightsquigarrow 6\times(5+1)$

D'où trois autres relations sur A :

- La clôture symétrique $\leftrightsquigarrow$ de $\rightsquigarrow$

  $6\times(5+1) \leftrightsquigarrow (2+4)\times(5+1)$

- La clôture réflexive-transitive $\rightsquigarrow^*$ de $\rightsquigarrow$

  $(2+4)\times(5+1) \rightsquigarrow^* 6\times 6$

- La relation d'équivalence $\leftrightsquigarrow^*$ engendrée par $\rightsquigarrow$

  $(2+4)\times(5+1) \leftrightsquigarrow^* 30+6$

Terminaison et confluence

- $b \in A$ est en forme normale si
  il n'existe aucun c tel que $b \rightsquigarrow c$.
- Le système termine si
  il n'y pas de suite infinie $a_1 \rightsquigarrow a_2 \rightsquigarrow a_3 \rightsquigarrow \ldots$.
- Le système est confluent si

$$\begin{tikzcd} & a \arrow[dl, squiggly] \arrow[dr, squiggly] & \\ b & & c \end{tikzcd} \quad \Longrightarrow \quad \exists d \quad \begin{tikzcd} & a \arrow[dl, squiggly] \arrow[dr, squiggly] & \\ b \arrow[dr, squiggly] & & c \arrow[dl, squiggly] \\ & d & \end{tikzcd}$$

Proposition. Si le système termine et est confluent, alors tout $a \in A$ a une unique forme normale.

Les entiers naturels

NAT.

Les termes sur la grammaire

$t ::= 0|1|2|\ldots|t+t|t-t$

avec $G \rightsquigarrow H$ pour “effectuer une opération”

forment un système qui termine et qui est confluent,

les termes en forme normale sont $0, 1, 2, \ldots$

NAT(x).

Les termes sur la grammaire

$t ::= 0|1|2|\ldots|x|y|z|\ldots|t+t|t-t$

avec $G \rightsquigarrow H$ pour “effectuer une opération”

ou “utiliser une propriété des opérations”

forment un système qui ne termine pas.

Ex. $2+3 \rightsquigarrow 5$, $x+0 \rightsquigarrow x$, $x \times 0 \rightsquigarrow 0$,

$x+y \rightsquigarrow y+x \rightsquigarrow x+y \rightsquigarrow \ldots$ par commutativité.

Correction et complétude

Étant donnée une relation d'équivalence $\approx$ sur A
et un système de réduction $\rightsquigarrow$ sur A.

- $\rightsquigarrow$ est correct par rapport à $\approx$
  si $a \leftrightsquigarrow^* b \implies a \approx b$
  tout ce qu'on peut obtenir par réduction est "vrai".
- $\rightsquigarrow$ est complet par rapport à $\approx$
  si $a \approx b \implies a \leftrightsquigarrow^* b$
  tout ce qui est "vrai" peut être obtenu par réduction.

Un modèle de $(A, \rightsquigarrow)$ interprète
les termes par des fonctions et $\rightsquigarrow$ par $=$. On pose :
$t \approx u \iff$ dans tout modèle, t et u ont la même interprétation.

Théorème de Birkhoff. $t \leftrightsquigarrow^* u \iff t \approx u$.

Règles de réécriture et pas de réécriture

Ex. NAT. $G \rightsquigarrow H$ *ssi* on passe de G à H en effectuant une addition ou une multiplication. Soient deux règles de réécriture :

$P_+ : \forall x, y \in \mathbb{N},\ x + y \rightsquigarrow z$ où $z = x + y$ dans $\mathbb{N}$
$P_\times : \forall x, y \in \mathbb{N},\ x \times y \rightsquigarrow z$ où $z = x \times y$ dans $\mathbb{N}$

Alors $G \rightsquigarrow H$ *ssi* on passe de G à H en appliquant une règle $P = L \rightsquigarrow R$. C'est un pas de réécriture :

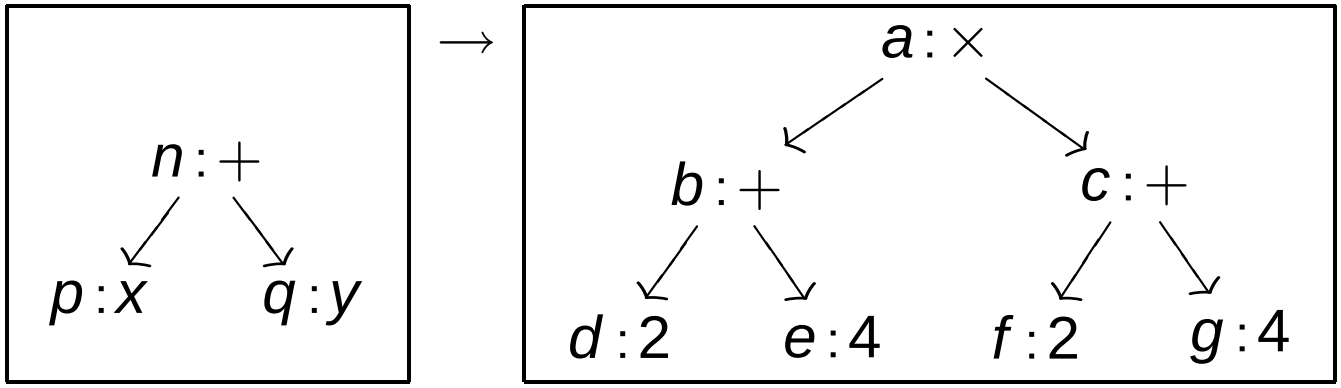
- trouver une instance de L dans G,
- remplacer cette instance de L par R,
- cela fournit H (avec une instance de R dans H).

Ex. NAT. $P_+ : x + y \rightsquigarrow z$ où $z = x + y$ dans $\mathbb{N}$
$G = (2 + 4) \times (5 + 1)$, avec $2 + 4$ instance de $x + y$ dans G
$G \rightsquigarrow H$ où $H = 6 \times (5 + 1)$, avec 6 instance de z dans H

Les termes vus comme des arbres

"$2+4$ instance de $x+y$ dans $(2+4)\times(2+4)$" ?
Pour éviter ces ambiguïtés, on peut voir un terme comme un arbre, donc un graphe. Alors une instance est un morphisme de graphes.

Ex.



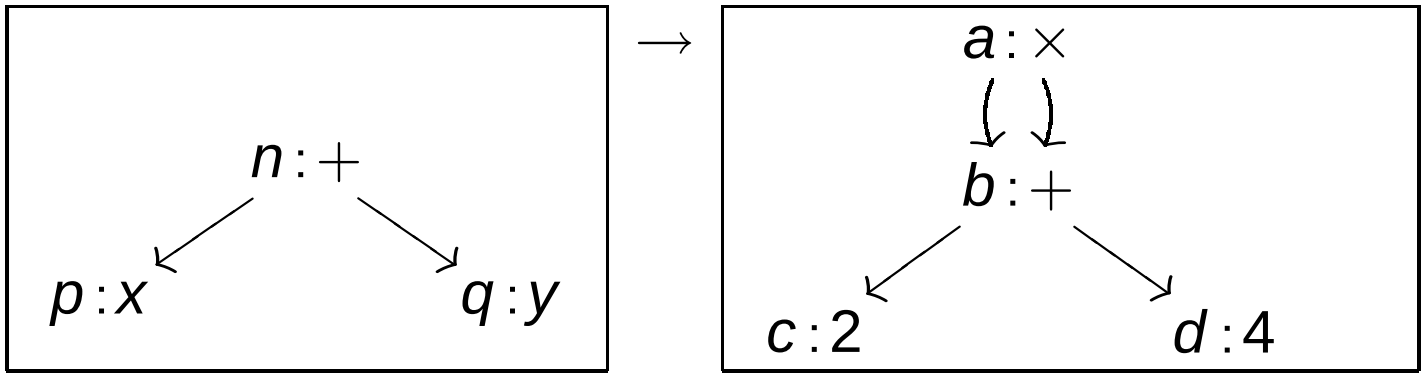
Deux instances de $x+y$ dans $(2+4)\times(2+4)$:

- le morphisme $n \mapsto b$, $p \mapsto d$, $q \mapsto e$, alors $H = 6 \times (2+4)$
- le morphisme $n \mapsto c$, $p \mapsto f$, $q \mapsto g$, alors $H = (2+4) \times 6$

Les termes vus comme des graphes acycliques

Généralisation : on peut voir un terme comme un graphe acyclique, pour grouper des sous-termes identiques.

Ex.



Une instance de $x + y$ dans $(2 + 4) \times (2 + 4)$:

- le morphisme $n \mapsto b$, $p \mapsto c$, $q \mapsto d$, alors $H = 6 \times 6$

Un cadre formel ?

À définir : objet, règle, instance, pas.

Facile.
Un objet est un graphe
Une instance est un morphisme de graphes
(Donc les objets et les instances forment une catégorie)

Difficile.
Une règle ce n'est *pas seulement* un couple d'objets,
ils doivent être "reliés entre eux"
et ensuite seulement on pourra définir un pas

Ex. $x + x \rightsquigarrow 2 \times x$: c'est bien le même x

En résumé...

IN :

$$\begin{array}{c} \boxed{L} \\ \downarrow \\ \boxed{\begin{array}{c} L \\ \hline G - L \end{array}} \end{array} \rightsquigarrow \boxed{R}$$

OUT :

$$\begin{array}{c} \boxed{L} \\ \downarrow \\ \boxed{\begin{array}{c} L \\ \hline G - L \end{array}} \end{array} \rightsquigarrow \begin{array}{c} \boxed{R} \\ \downarrow \\ \boxed{\begin{array}{c} R \\ \hline H - R \end{array}} \end{array}$$

- nature de $\rightsquigarrow$?
- nature de la construction ?

Outline

Introduction

Réécriture de termes

Réécriture de graphes

Conclusion

Graphes

Les graphes servent un peu partout,
la réécriture de graphes aussi. . .

- réécriture de termes “économique”
- bases de données
- démonstration automatique
- programmtion fonctionnelle
- prototypage
- gestion de mémoire

Il y a différentes sortes de graphes
et différentes sortes de réécriture de graphes. . .
Ici : DPO, DPOred, SPO, HPO
où PO = Pushout = somme amalgamée

Union d'ensembles

$$\begin{tikzcd}
K = R \cap D \arrow[r, "\subseteq"] \arrow[d, "\subseteq"'] \arrow[dr, phantom, "="] & R \arrow[d, "\subseteq"] \\
D \arrow[r, "\subseteq"'] & H = R \cup D
\end{tikzcd}$$

Ce carré est un pushout :

pour tout carré commutatif
$$\begin{tikzcd}
R \cap D \arrow[r, "\subseteq"] \arrow[d, "\subseteq"'] \arrow[dr, phantom, "="] & R \arrow[d] \\
D \arrow[r] & H'
\end{tikzcd}$$

il existe un unique $h : H = R \cup D \to H'$ tel que

$$\begin{tikzcd}
R \cap D \arrow[r, "\subseteq"] \arrow[d, "\subseteq"'] \arrow[dr, phantom, "="] & R \arrow[d, "\subseteq"] \arrow[ddr] & \\
D \arrow[r, "\subseteq"'] \arrow[drr] & R \cup D \arrow[dr, "="] & \\
& & H'
\end{tikzcd}$$

De plus : $\mathrm{Card}(H) - \mathrm{Card}(D) = \mathrm{Card}(R) - \mathrm{Card}(K)$.

Pushout d'ensembles

$$\begin{tikzcd} K \arrow[r, "r"] \arrow[d, "d"'] \arrow[dr, phantom, "="] & R \arrow[d] \\ D \arrow[r] & H \end{tikzcd}$$

Ce carré est un pushout *ssi*

$$H = (R + D)/\sim \quad \text{où :}$$

- $R + D$ est l'union disjointe de R et D,
- $\sim$ est la relation d'équivalence engendrée par

$$\forall x \in K,\ d(x) \sim r(x)$$

1. Double Pushout

[Ehrig et al. 73]

- Un objet est un graphe
- Une instance est un morphisme de graphes $\begin{tikzcd} L \arrow[d, "m"'] \\ G \end{tikzcd}$
- Une règle est un span de graphes $\begin{tikzcd} L & K \arrow[l, "l"'] \arrow[r, "r"] & R \end{tikzcd}$
- Un pas est un double pushout de graphes :

$$\begin{tikzcd} L \arrow[d, "m"'] & K \arrow[l, "l"'] \arrow[r, "r"] \arrow[d] & R \arrow[d] \\ G & D \arrow[l] \arrow[r] & H \end{tikzcd}$$

K est “la partie commune” à L et R,
D est “la partie commune” à G et H.
Typiquement, l est injectif.

Á gauche : pushout complément

Étant donné
$$\begin{tikzcd} L \arrow[d, "m"'] & K \arrow[l, "l"'] \\ G & \end{tikzcd}$$
trouver
$$\begin{tikzcd} L \arrow[d, "m"'] \arrow[dr, phantom, "="] & K \arrow[l, "l"'] \arrow[d, "d"] \\ G & D \arrow[l] \end{tikzcd}$$

tel que pour tout carré commutatif
$$\begin{tikzcd} L \arrow[d] \arrow[dr, phantom, "="] & K \arrow[l, "l"'] \arrow[d, "d"] \\ G' & D \arrow[l] \end{tikzcd}$$

il existe un unique $g : G \to G'$ tel que

$$\begin{tikzcd} L \arrow[d] \arrow[dr, phantom, "="] \arrow[ddl] & K \arrow[l, "l"'] \arrow[d, "d"] \\ G \arrow[d, phantom, "="] & D \arrow[l] \arrow[dl] \\ G' & \end{tikzcd}$$

Le pushout complément, s'il existe, n'est pas unique.

Á droite : pushout

Étant donné $\begin{tikzcd} K \arrow[r, "r"] \arrow[d, "d"'] & R \\ D & \end{tikzcd}$ trouver $\begin{tikzcd} K \arrow[r, "r"] \arrow[d, "d"'] \arrow[dr, phantom, "="] & R \arrow[d] \\ D \arrow[r] & H \end{tikzcd}$

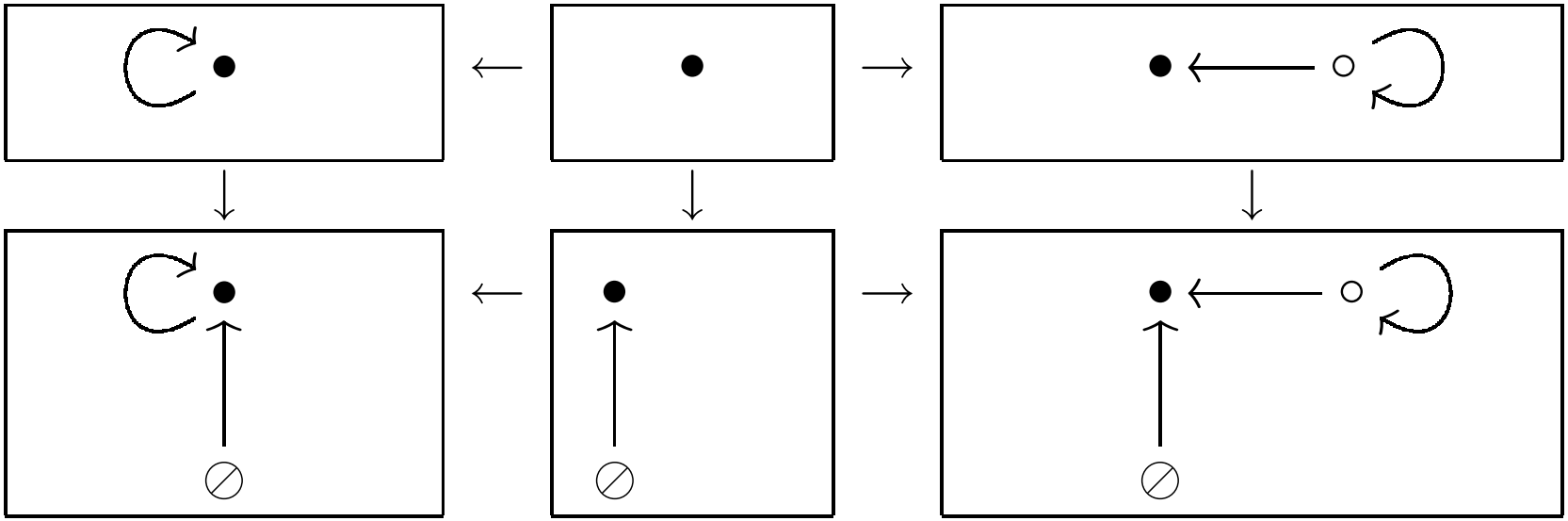
tel que pour tout carré commutatif $\begin{tikzcd} K \arrow[r, "r"] \arrow[d, "d"'] \arrow[dr, phantom, "="] & R \arrow[d] \\ D \arrow[r] & H' \end{tikzcd}$

il existe un unique $h : H \to H'$ tel que

$$\begin{tikzcd} K \arrow[r, "r"] \arrow[d, "d"'] \arrow[dr, phantom, "="] & R \arrow[d] \arrow[ddr, "="] & \\ D \arrow[r] \arrow[drr] & H \arrow[dr, "="] & \\ & & H' \end{tikzcd}$$

Le pushout, s'il existe, est unique.

DPO : un exemple



2. Double Pushout “avec redirection”

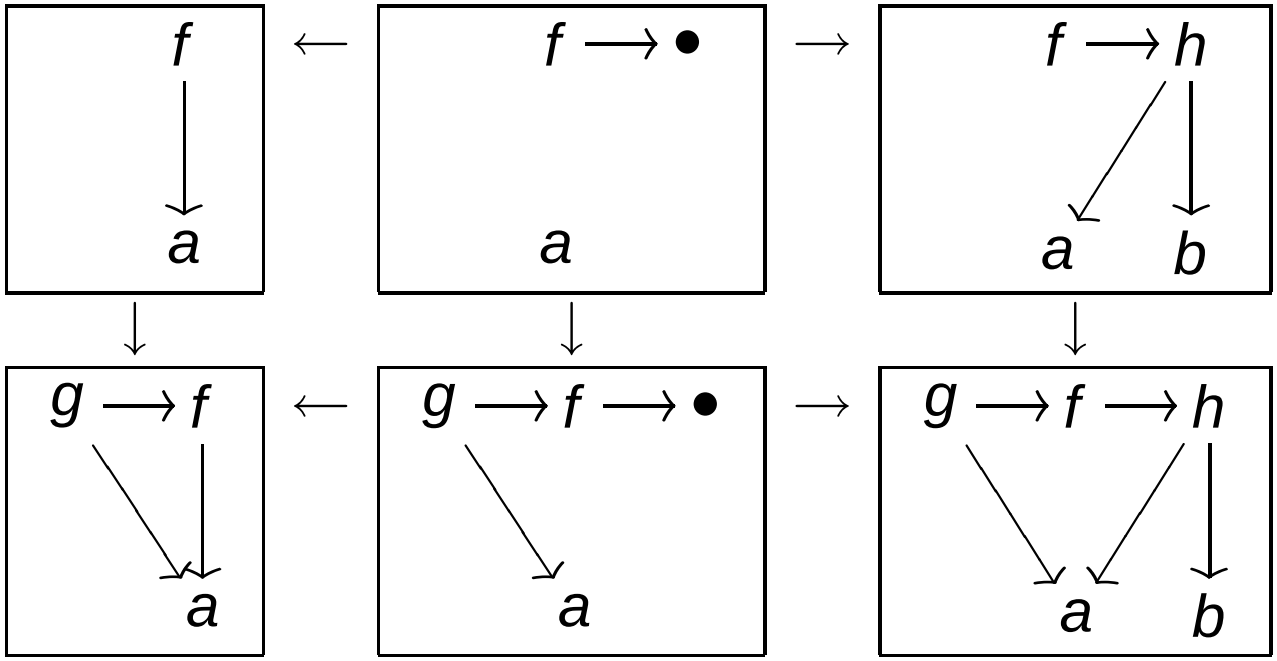
[DEP 07]
Comme pour le DPO classique
un pas est un double pushout de graphes :

$$\begin{tikzcd} L \arrow[d, "m"'] & K \arrow[l, "l"'] \arrow[r, "r"] \arrow[d] & R \arrow[d] \\ G & D \arrow[l] \arrow[r] & H \end{tikzcd}$$

K est obtenu en “décrochant” des arcs de L,
puis R en “raccrochant” ces arcs
De même pour D.

Typiquement, l est surjectif.

DPO avec redirection : un exemple



Remarques

— Pour le DPO avec redirection :

- chaque nœud n est étiqueté par une opération $n : f$ éventuellement par une variable $n : \bullet$
- chaque opération f a une arité fixée dans $\mathbb{N}$
- les successeurs d'un noeud $n : f$ représentent les arguments de f : il y en a n et ils sont ordonnés

Cela représente l'état d'une mémoire
avec les cellules (nœuds) et les pointeurs (arcs).

Proposition. Dans ce cadre, un morphisme de graphes est déterminé par l'image des nœuds.

— Pour les DPO's en général : pas de ramasse-miettes.

3. Simple Pushout

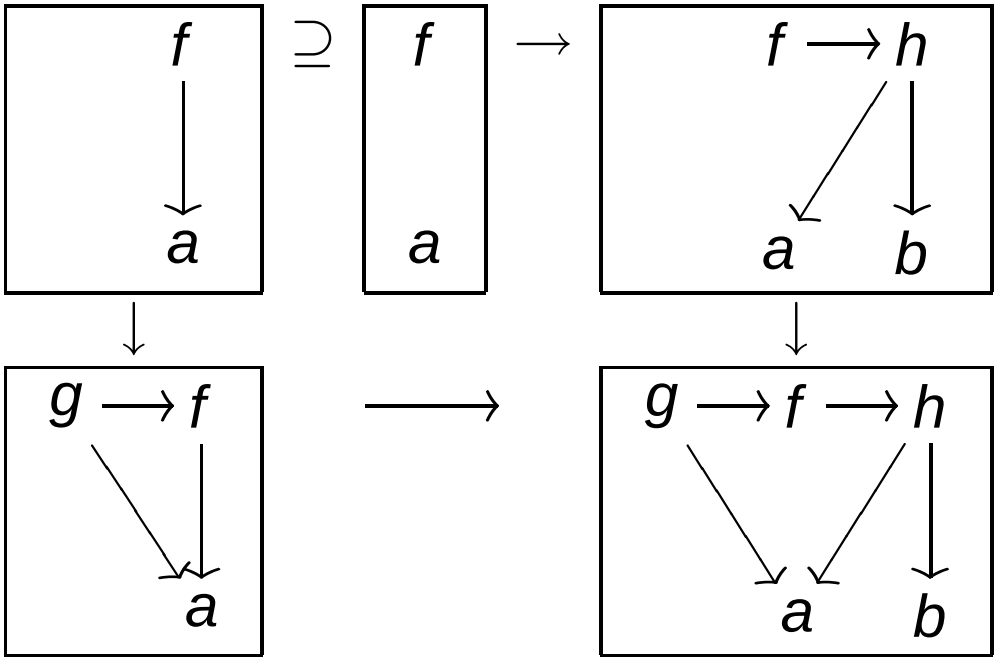
[Raoult 84, Kennaway 87]

- Un objet est un graphe
- Une instance est un morphisme de graphes $\begin{array}{c} L \\ {\scriptstyle m}\downarrow \\ G \end{array}$
- Une règle est un morphisme partiel de graphes $L \xrightarrow{p} R$
- Un pas est un simple pushout de graphes :

$$\begin{array}{ccc} L & \xrightarrow{p} & R \\ {\scriptstyle m}\downarrow & & \downarrow \\ G & \longrightarrow & H \end{array}$$

$\mathrm{Dom}(p)$ est “la partie à recopier” de L.

SPO : un exemple



4. Pushout Hétérogène

[DEP 09]

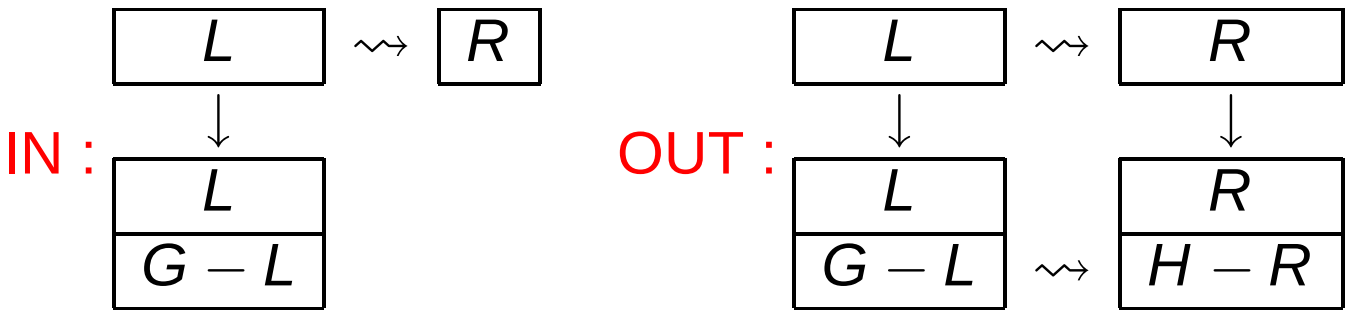
Conserver “nos” graphes avec leurs propriétés :

- représenter l’état d’une mémoire avec les cellules et les pointeurs
- un morphisme de graphes est déterminé par l’image des nœuds

et de plus :

- prendre en compte le ramasse-miettes

Rappel. . .

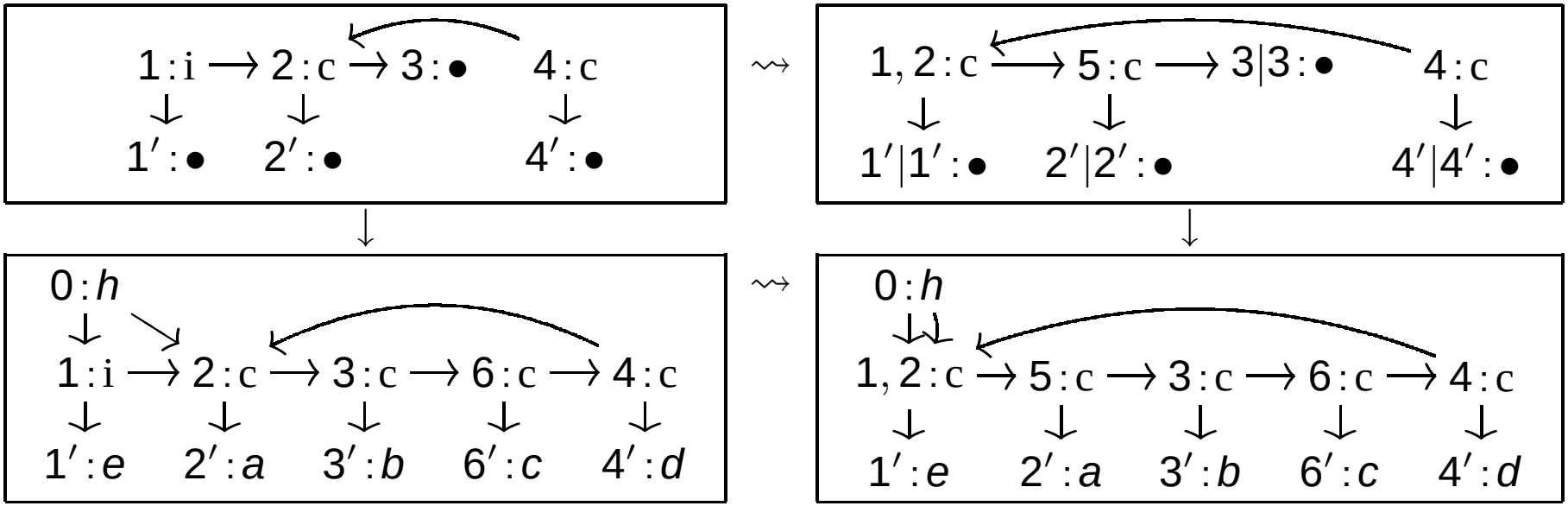


Que faire des flèches de G :

- de $G - L$ vers L ?
  il faut les raccrocher quelque part
- de L vers $G - L$, ou de L vers L mais pas dans L ?
  on peut les “cloner”

HPO : un exemple

Insérer un élément dans une liste circulaire (i=insert, c=cons)



Bilan : le nombre de cellules n'a pas changé.

Théorème. $\mathrm{Card}(H) - \mathrm{Card}(G) = \mathrm{Card}(R) - \mathrm{Card}(L)$.

Outline

Introduction

Réécriture de termes

Réécriture de graphes

Conclusion

- Par ailleurs, en logique, la déduction peut être vue comme une réécriture, qui utilise encore des pushouts. . .
  mais fondé sur la théorie des catégories de fractions de Gabriel-Zisman (1967), utilisée en topologie algébrique.
- Collaboration avec Rachid et Frédéric : des projets...

Merci pour votre attention